DRAFT PAPER TO BE PRESENTED AT THE 7TH INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH. FOR CIRCULATION AT THIS SYMPOSIUM ONLY.

A Random Sampling Scheme for Path Planning

Jérôme Barraquand* Lydia Kavraki† Jean-Claude Latombe†
Tsai-Yen Li‡ Rajeev Motwani§ Prabhakar Raghavan¶

Abstract

Several randomized path planners have been proposed over the last few years. Their attractiveness stems from their applicability to virtually any type of robots, and their empirically observed success. In this paper we attempt to present a unifying view of these planners and to theoretically explain their success. First, we introduce a general planning scheme that consists of randomly sampling the robot's configuration space. We then describe two previously developed planners as instances of planners based on this scheme, but applying very different sampling strategies. These planners are probabilistically complete, that is: if there exists a path, the planner will find it with high probability, if we let it run long enough. Next, for one of the two planners, we analyze more formally the relation between the probability of failure and the running time. Under two different assumptions about the robot's free space, we show that the running time only grows as the logarithm of the probability of failure that we are willing to tolerate. In the last section of the paper we suggest directions of future research.

*Salomon Brothers Int. Ltd., Victoria Plaza, 111 Buckingham Palace Road, London SW1W 0SB, UK.
†Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305, USA.
‡Computer Science Department., National Chenchi University, Wenshan, Taipei, Taiwan.
§Department of Computer Science, Stanford University, Stanford, CA 94305, USA.
¶IBM Almaden Research Center, San Jose, CA 95120, USA.

1 Introduction

Robot path planning has been proven a hard problem [28]. There is strong evidence that its solution requires exponential time in the number of dimensions of the configuration space, i.e., the number of degrees of freedom (dofs) of the robot. This result is remarkably stable; indeed, it still holds for rather specific robots, e.g., planar linkages consisting of links serially connected by revolute joints [11] and sets of rectangles executing axis-parallel translations in a rectangular workspace [8, 9]. Though general and complete algorithms have been proposed [5, 30], their high complexity precludes any useful application. This negative result has led some researchers to seek heuristic algorithms. While several such algorithms may solve apparently difficult problems, they also often fail or take prohibitive time on seemingly simpler ones. The fact that their behavior is not well characterized is a major drawback when they are used as blackboxes in larger robot control systems.

The number of dofs beyond which complete algorithms become practically useless is low, somewhere between 3 and 5. This means that they cannot be used to compute paths for rigid objects translating and rotating in three dimensions, nor for six-dof manipulator arms, two important cases in practice. On the other hand, robot applications tend to involve more degrees of freedom than ever before. For example, an increasing number of manufacturing workcells use several cooperating robots to augment throughput and flexibility. Cells with more than twenty dofs are no longer exceptions. As costs and time for designing and deploying them become more critical, path planners integrated with CAD systems will be in higher demand to facilitate robot programming. Eventually, planners will run online to allow for non-deterministic sequences of goals and events [23]. Robots in domains other than manufacturing (e.g., medical surgery, space exploration) will also require efficient and reliable path planners. Some non-robotics domains raise a similar need as well. In computer graphics, animation of synthetic actors to produce digital movies or video games requires dealing with several dozen dofs. Here, motion planning may drastically reduce the work of human animators who currently input large numbers of key frames. In molecular biology, motion planning can help compute plausible docking motions of molecules modeled as spatial linkages with many dofs.

Collision-free path planning, which typically assumes perfect knowledge of the world and stationary obstacles, is only the most basic motion planning problem in robotics. Clearly, we would ultimately like robot planners to also deal with issues such as uncertainties, moving obstacles, movable objects, and dynamic constraints [22]. But every extension of the basic problem adds in computational complexity. For instance, allowing moving obstacles makes the problem exponential in the number of moving obstacles [5, 29]; uncertainties in control and sensing makes the problem exponential in the complexity of the robot environment [5]. Before we can effectively investigate such extensions in large configuration spaces, it seems that we must better understand how to practically solve basic path planning.

Path-planning applications are so diverse that it is infeasible to design a tailor-made al-

gorithm for every possible robot.[1] Instead, we need general path planning algorithms not bound to the specifics of any particular robot. We believe that between the two extreme types of planners suggested above – complete and heuristic – there is place for practically efficient general planners achieving a weaker form of completeness. In other words, we may perhaps trade a limited amount of completeness against a major gain in computing efficiency. Full completeness requires the planner to always answer a path-planning query correctly, in asymptotically bounded time. A weaker, but still interesting form of completeness is the following: if a solution path exists, the planner will find one in bounded time, with high probability. We call it *probabilistic completeness*.

With this philosophy in mind, we have designed a series of new path planners and experimented with them in large configuration spaces. One of them, described in [2, 3, 22], is a potential-field-based planner that escapes local minima by performing random walks; in the following, we will refer to it as the *potential-field planner*. Another planner, presented in [14, 15, 17], precomputes a "roadmap" (network) of simple paths connecting randomly selected configurations and tries to construct a path between any two input configurations by connecting them to this roadmap; we will refer to this planner as the *roadmap planner*. Both these planners have been successfully applied to complex problems. For example, in [19], the potential-field planner was used to automatically synthesize a video clip with graphically simulated human and robot characters entailing a configuration space with 78 dimensions. Both the potential-field and the roadmap planners have been used to check that some parts can be removed from an aircraft engine for inspection and maintenance [7]; here, paths are generated in configuration spaces having only six dimensions, but the geometry of the workspace is particularly complex.

These two planners achieve some form of probabilistic completeness. For the potential-field planner, this property remains qualitative: if there exists a path, the probability that the planner finds it tends toward one as the running time increases; but the convergence speed is unknown. On the other hand, for variants of the roadmap planner, we have proven stronger results that relate the probability that it finds a path, when one exists, to its running time [13, 16]. In turn, these theoretical results suggest improvements, which are being considered in our current research.

Other work investigating similar or related randomized planning approaches include [1, 4, 10, 25, 26].

This paper proposes a consistent framework to describe and study the randomized planners cited above, with the goal to eventually build more powerful planners. In the planners cited above, the robot's free space is not explicitly represented, but randomly sampled. We believe that this is the central concept underlying their success. In Section 1 we capture this concept into a general computational scheme for path planning in large configuration spaces.[2] In

[1] In any case, very few tailor-made planners have been successfully designed for specific robots with more than four dofs.

[2] This scheme can also be applied to configuration spaces having few dimensions; but it is less interesting

Section 2 we make our discussion more precise by presenting our potential-field and roadmap planners as two instances of planners using this scheme, but applying two different sampling strategies. In Section 3 we give two formal analyses of the probabilistic completeness of variants of the roadmap planner. This analysis work is still at an early stage, but we hope that its results will help producing more throrough analyses in the future. Indeed, as it has become relatively easy to design new randomized path planners that perform better than previous ones on some well-selected problems, it is increasingly important to formally analyze these planners to better explain their successes and failures.

This paper does not present experimental results obtained with our planners. Many such results have already been reported in [2, 7, 14, 15, 17, 19, 20], as well as in papers by other authors [6, 32].

2 General Scheme

We consider the problem of planning collision-free paths for an arbitrary n-degree-of-freedom holonomic[3] robot $\mathcal{A}$. We let $\mathcal{C}$ denote the configuration space of $\mathcal{A}$ and $\mathcal{C}_{free}$ stand for the open subset of collision-free configurations in $\mathcal{C}$. We also refer to $\mathcal{C}_{free}$ as the *free space* and to configurations in $\mathcal{C}_{free}$ as *free configurations*. A planning problem is specified by two free configurations, $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, called the *initial* and the *goal* configurations, respectively. Any path lying in $\mathcal{C}_{free}$ that joins these two configurations is a solution of this problem.

In the following, we will assume for simplification that $\mathcal{C}$ is the cube $[0,1]^n$, so that each configuration $\mathbf{q}$ is described by an n-tuple $(q_1, \ldots, q_n)$. However, it is straightforward to extend our presentation to cases where $\mathcal{C}$ is multiply connected, i.e., one or more dimensions can "wrap around". The only required assumption is that $\mathcal{C}$ be measurable, which is always achieved in practice.

Although there exist algorithms to construct an explicit representation of $\mathcal{C}_{free}$ given the geometry of $\mathcal{A}$ and the obstacles in semi-algebraic form, their time complexity makes it impractical for any sufficiently large value of n. On the other hand, reasonably efficient algorithms are available to compute the Euclidean distance between two objects in three-dimensional space (e.g., [27]). This leads us to assume that $\mathcal{C}_{free}$ is implicitly given by a function, CLEARANCE: $\mathcal{C} \rightarrow \mathbb{R}$, that maps any configuration to the distance between the robot (placed at that configuration) and the obstacles, if there is no contact between them, and to 0 (or a negative number), otherwise. Thus, whenever CLEARANCE($\mathbf{q}$) is positive, $\mathbf{q}$ belongs to $\mathcal{C}_{free}$. We refer to CLEARANCE($\mathbf{q}$) as the *clearance* of $\mathbf{q}$.

For any bounded robot $\mathcal{A}$, there exists a constant ρ such that when $\mathcal{A}$ moves along a straight path in $\mathcal{C}$ between any two configurations $\mathbf{q} = (q_1, \ldots, q_n)$ and $\mathbf{q}' = (q'_1, \ldots, q'_n)$, no point of $\mathcal{A}$ traces a curve segment longer than $\rho \max_{i \in [1,n]} |q_i - q'_i|$. We assume here that ρ is given,

in that case, since complete planning techniques are then available.

[3] Our presentation can be extended to nonholonomic robots. See [26, 31].

though for most usual robots its computation is straightforward and can be included in the planner. Rather than using a single ρ, we may also partition $\mathcal{C}$ into several regions and have a distinct constant ρ_i for each region. If the ρ_i's differ by large amounts, this could substantially reduce the running time of a planner based on the scheme presented below, but we will not go into such detail in this paper.

Definition 1 *Let* $\mathbf{q} = (q_1, \ldots, q_n)$ *and* $\mathbf{q}' = (q'_1, \ldots, q'_n)$ *be two free configurations whose respective clearances are* η *and* η'. *They are said to be adjacent if* $\rho \max_{i\in[1,n]} |q_i - q'_i| < \max\{\eta, \eta'\}$.

Note that $\max_{i\in[1,n]} |q_i - q'_i|$ defines a distance in $\mathcal{C}$ (the L^∞ metric). We could have used any other distance in $\mathcal{C}$ to define adjacency. Of course, the constant ρ depends on the chosen distance.

Clearly, if two configurations are adjacent the straight-line segment joining them lies entirely in $\mathcal{C}_{free}$.

Now we can state our **general planning scheme** as follows:

> Pick configurations in $\mathcal{C}$ at random; retain those configurations which lie in $\mathcal{C}_{free}$ (along with $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$) as the nodes of a graph G; and connect adjacent configurations by links of G.
>
> Return YES as soon as $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ belong to the same connected component of G.
>
> Return NO if no path has been found after having generated c configurations, where c is an input parameter.

The answer YES is always correct, that is, whenever the planner returns YES, there actually exists a collision-free path connecting $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$; moreover such a path can easily be extracted from G. On the other hand, the answer NO is not necessarily correct. Indeed, after a finite amount of computation (defined by the parameter c), the planner may still not have found a path between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, even if one exists.

The key component of this planning scheme is the *sampling strategy* applied to generate the nodes of G. Different strategies are possible. For example, if several planning queries are to be made for the same robot with the same obstacles (*multi-query case*), then it may be suitable to invest some preprocessing time in order to construct a network of configurations (we call such a network a *roadmap*). Processing each query then only requires connecting the input initial and goal configurations to the roadmap. If the queries are not known in advance, it seems reasonable to construct the roadmap by choosing configurations uniformly at random from $[0, 1]^n$ since the resulting free configurations are then uniformly distributed over $\mathcal{C}_{free}$. However, we will see in Section 3.2 that, while the roadmap is being generated, it is possible to derive heuristic information from it and bias the selection of the new configurations.

Instead, if a single or very few queries are to be made with the same robot/obstacles (*single-query case*), the planner may be more successful by using a sampling strategy that picks new configurations, first in neighborhoods of the initial and goal configurations and then, iteratively, in neighborhoods of the newly generated configurations, until the two "sampling waves" meet. A similar strategy could also be used in the multi-query case to connect the initial and goal configurations to the roadmap.

Although the efficiency of any planner may be evaluated through experimentation, formal analysis is desirable to compare different planners and understand their strengths and weaknesses. Ideally a planner's outcome should be YES with the highest probability whenever a free path exists and this outcome should be generated in minimal time. Since this goal combines two subgoals that are usually conflicting, the analysis should relate the probability that the planner produces an incorrect answer to its running time. Roughly speaking, given a small probability (the acceptable probability that the planner's output is NO while a free path exists), we would like to bound the running time of the planner by a function of this probability.

Let us assume for simplification that CLEARANCE takes constant time to evaluate. Then the planner's running time mainly depends on two numbers: the number of sample configurations and the number of pairs of free configurations checked for adjacency. The number of sample configurations is at most c. But only a fraction of them, f, are in $\mathcal{C}_{free}$, hence in the constructed graph G. In practice, it is often the case that $f \ll f$. The planner should thus strive to get the greatest ratio f/c, but the distribution of the generated free configurations is also crucial to eventually find a path. The number of adjacency checks is at most proportional to f^2; however, the planner may choose not to test all pairs of free configurations for adjacency.

It seems likely that no strong property can be proven for any given planner, if we do not make some assumption about $\mathcal{C}_{free}$. Moreover, no single planner is likely to be the most efficient for all possible problems. This suggests that planners should be analyzed under some well-specified assumptions. In Section 4, we will study the work carried out by a two-phase planner under two distinct assumptions. In one, the *visibility volume assumption*, $\mathcal{C}_{free}$ is such that every free configuration "sees" a subset of $\mathcal{C}_{free}$ whose volume is at least an ϵ fraction of the total volume of $\mathcal{C}_{free}$ (we then say that $\mathcal{C}_{free}$ is ϵ-good). In the second assumption, the *path clearance assumption*, there exists a collision-free path between $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ that has some given clearance ξ.

Assumptions must be carefully crafted. Indeed, if these are too specific or unrealistic, the analysis will not yield useful results; on the other hand, if they are too general, the results will be too weak.

3 Specific Planners

In this section we present two specific planners that make use of the above scheme with different sampling strategies: the potential-field and the roadmap planners.

3.1 Potential-Field Planner

This planner was originally described in [2, 3, 22], along with experimental results.[4] Extensions and additional experimental results were presented in [7, 19, 20]. Here, we focus on its relation to the general scheme presented above.

The planner is given a function $\mathbf{U} : \mathcal{C}_{free} \rightarrow \mathbb{R}^+ \cup \{0\}$, the *potential field*, with a single global minimum (0) at the goal configuration.[5] It then attempts to connect $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ by alternating *down motions* and *escape motions*. Each down motion "descends" along $\mathbf{U}$ until it reaches a local minimum, while escape motions attempt to flee from local minima. More precisely:

The following algorithm constructs a down motion starting at configuration $\mathbf{q}_s$ (h is an input parameter):

`Down-Motion`($\mathbf{q}_s$):

1. $\mathbf{q} \leftarrow \mathbf{q}_s$.
2. Until $\mathbf{q}$ is not labelled as a local minimum do:
   (a) Pick at random up to h configurations adjacent to $\mathbf{q}$, until one of them, $\mathbf{q}'$, satisfies $\mathbf{U}(\mathbf{q}') < \mathbf{U}(\mathbf{q})$.
   (b) If the previous step succeeded in generating $\mathbf{q}'$, then reset $\mathbf{q}$ to $\mathbf{q}'$; else label $\mathbf{q}$ as a local minimum.
3. Return $\mathbf{q}$.

At Step 2(a) let $\mathbf{q} = (q_1, \ldots, p_n)$ and $\eta = \text{CLEARANCE}(\mathbf{q})$. Configurations are picked at random from the volume defined by $\Pi_{i=1,\ldots,n}[q_i - \eta/\rho, q_i + \eta/rho] \cap [0,1]^n$. According to Definition 1, all configurations in this volume are adjacent to $\mathbf{q}$.

To guarantee that Step 2 does not loop for ever, $\mathbf{U}$ must have no local minimum in the boundary of $\mathcal{C}_{free}$. This can easily be obtained by including in the definition of $\mathbf{U}(\mathbf{q})$ a term that is proportional to $1/\text{CLEARANCE}(\mathbf{q})$.

The following algorithm constructs an escape motion starting at configuration $\mathbf{q}_l$ (a local minimum):

[4] The planner is available by anonymous ftp from `flamingo.stanford.edu:/pub/li/rpp3d.tar.gz`.
[5] Techniques to automatically construct $\mathbf{U}$ are proposed in [2, 22].

`Escape-Motion`($\mathbf{q}_l$):

1. Pick at random the number of steps, m, of the motion.[6]
2. $\mathbf{q} \leftarrow \mathbf{q}_l$.
3. For $i = 1, \ldots, m$ do:
   (a) Pick at random a free configuration $\mathbf{q}'$ adjacent to $\mathbf{q}$.
   (b) $\mathbf{q} \leftarrow \mathbf{q}'$.
4. Return $\mathbf{q}$.

The overall planning algorithm is the following:

`Potential-Field-Planner`($\mathbf{q}_{init}, \mathbf{q}_{goal}$):

1. $\mathbf{q}_l \leftarrow$ `Down-Motion`($\mathbf{q}_{init}$).
2. While $\mathbf{q}_l \neq \mathbf{q}_{goal}$ do:
   (a) Do:
      i. If the total number of configurations generated so far is greater than c then return NO and halt.
      ii. $\mathbf{q}_s \leftarrow$ `Escape-Motion`($\mathbf{q}_l$);
      iii. $\mathbf{q}'_l \leftarrow$ `Down-Motion`($\mathbf{q}_s$);

   until $\mathbf{U}(\mathbf{q}'_l) < \mathbf{U}(\mathbf{q}_l)$.
   (b) $\mathbf{q}_l \leftarrow \mathbf{q}'_l$.
3. Return YES.

The graph G constructed by the potential-field planner is a tree of paths rooted at $\mathbf{q}_{init}$. Each configuration is generated such that it is adjacent to a previously generated configuration, and is only connected to this configuration. Hence, the planner does not test all pairs of configurations for adjacency, avoiding the quadratic-time cost of this test. During a down motion, the potential $\mathbf{U}$ introduces a bias in the choice of the tree path that will be followed by the planner. Although the global geometry of $\mathcal{C}_{free}$ is unknown, $\mathbf{U}$ can be seen as a *specialist* that gives some heuristic indications about this geometry, i.e.: which directions are promising and which aren't. The techniques in [2, 22] compute $\mathbf{U}$ by combining local-minima functions computed over the robot's workspace. Although the resulting $\mathbf{U}$ often prevents the

[6] In [2] we suggest to choose m according to a truncated Laplace distribution with mean value 1 (the "radius" of the configuration space). The intuition is that an escape motion should rarely be much longer than 1 to succeed.

planner from getting trapped into big obstacle cavities, it still has local minima and therefore is an imperfect characterization of the free space's geometry. On the other hand, computing local-minima-free potentials is a difficult problem [18] that is at least as hard as path planning itself.

Using well-known properties of random motions, the potential-field planner can be shown probabilistically complete [2, 21]. But no analytical result has been established quantifying how fast the probability that the planner returns NO converges toward zero as the running time increases. In fact, the use of a heuristic potential seems a major obstacle toward obtaining such a result. Despite this lack of analytical results, the planner has solved many difficult problems. On the other hand, it is not difficult to create planning problems that it fails to solve in a reasonable amount of time, though they admit rather obvious solutions.

The failures of the potential-field planner are often caused by *traps*. We define a trap as a basin of attraction of a local minimum of $\mathbf{U}$ that is almost completely surrounded by forbidden configurations (i.e., $\mathcal{C}\backslash\mathcal{C}_{free}$). Hence, a trap admits only narrow exits and the potential function inside the trap guides the robot away from these exits. If the robot's path starts within a trap or is guided into one by $\mathbf{U}$, each escape motion executed then has a tiny probability of escaping the local minimum. One could imagine a variant of the planner where the potential field is randomly guessed among a collection of several functions and changed several times during planning, hoping that the various potentials would entail different traps. We did some experiments with this variant, but we got no significant results.

3.2 Roadmap Planner

The problems encountered with the potential-field planner led us to develop another randomized approach to path planning [12, 14, 15, 17]. Several variants of this approach have been implemented. The planner presented below corresponds to the variant described in [12, 17]. Unlike the potential-field planner, the roadmap planner uses no problem-specific heuristics.

The roadmap planner operates in two phases, preprocessing and query processing: The preprocessing phase consists of constructing a network R of configurations, the roadmap. The configurations in R are called *milestones*. They only form a subset of all the sample configurations generated by the planner; hence, the roadmap is not exactly the graph G mentioned in the general scheme. Every query specifies two configurations, $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$, in $\mathcal{C}_{free}$. Processing the query consists of connecting these configurations to two milestones and checking that these two milestones are in the same connected component of R.

The planner uses a simple algorithm, called the *connector*, to construct the links of R. Given any two milestones, $\mathbf{m}_1$ and $\mathbf{m}_2$, the connector checks if the straight line segment connecting them lies in $\mathcal{C}_{free}$. It does so by recursively breaking the line segment into shorter segments and checking the endpoints of each segment for adjacency, until either one endpoint is not in free space or a sequence of adjacent configurations has been found between the two milestones. In the second case, the connector generates a link between $\mathbf{m}_1$ and $\mathbf{m}_2$ in the roadmap. The

work done by connector is part of the sampling work done by the roadmap planner, but the sample configurations it generates are not permanently stored; they are not part of the roadmap R. The connector could try several canonical paths, rather than just straight ones. To keep our presentation simple, however, we will assume that it only tries straight paths. In the following we say that a configuration *sees* another configuration if the straight-line segment joining them lies entirely in $\mathcal{C}_{free}$.

In the following algorithm we could limit the total number of sample configurations to c, as in the general scheme of Section 2. However, putting the limit on the number of generated milestones instead is slightly more convenient; this will also faciltate the analyses of Section 4. The preprocessing phase constructs an initial roadmap containing r milestones (Steps 1, 2, and 3 in the algorithm shown below). Then it expands this initial roadmap into a final one containing $s > r$ milestones. Both r and s are input parameters. The first r milestones are chosen uniformly over $\mathcal{C}_{free}$. The remaining $s - r$ milestones are selected in small regions considered as "difficult" parts of $\mathcal{C}_{free}$. The preprocessing algorithm is the following:

`Preprocessing`:

1. $i \leftarrow 0$.
2. While $i < r$ do:
   (a) Pick a configuration $\mathbf{q}$ in $\mathcal{C}$ at random.
   (b) If CLEARANCE($\mathbf{q}$) > 0 then
      i. Store $\mathbf{q}$ as a milestone of the roadmap.
      ii. $i \leftarrow i + 1$.
3. For every pair of milestones $\mathbf{m}_1$ and $\mathbf{m}_2$ whose distance is less than d do:
   (a) If $\mathbf{m}_1$ and $\mathbf{m}_2$ see each other then insert a link between them in the roadmap.
   (b) Update the roadmap's connected components.
4. Invoke `Resample` to expand the roadmap by $s - r$ milestones (see below).

The threshold d is an input parameter. It is used to limit the number of pairs of milestones that are tested by the connector, since in most spaces two milestones that are far apart are unlikely to see each other. Step 3 takes time at most quadratic in r, which is usually much smaller than the total number of sample configurations generated by the roadmap planner.

The `Resample` algorithm selects milestones with probability proportional to a "failure ratio" $R_f(\mathbf{m})$ computed at Step 3 and defined as follows:

$$R_f(\mathbf{m}) = \frac{F(\mathbf{m})}{E(\mathbf{m}) + 1},$$

where $E(\mathbf{m})$ is the total number of times the connector tried to link $\mathbf{m}$ to another milestone and $F(\mathbf{m})$ is the number of times it failed. Intuitively, if this ratio is large, $\mathbf{m}$ lies in a difficult region of $\mathcal{C}_{free}$. But this is only a heuristics and other definitions of R_f could be used as well. For each milestone $\mathbf{m}$ that it selects, `Resample` picks a number of new milestones at random in a neigborhood centered at $\mathbf{m}$ and invokes the connector to try to link each of these new milestone to other milestones that are distant by less than d. This expansion of the roadmap terminates when the total number of milestones is s.

Experiments have been done with and without the roadmap expansion step. Roadmaps containing the same total number of milestones have been constructed both ways. The roadmaps generated using the expansion step have been consistently better, i.e., subsequent queries were processed more quickly with less false NO answers. Over a large range of problems, generating 2/3 of the milestones during the initial step and 1/3 during the expansion step gave good results.

After preprocessing, each query is handled as follows (g is an input parameter):

`Query-Processing`$(\mathbf{q}_{init}, \mathbf{q}_{goal})$:

1. For $i = \{init, goal\}$ do:
   (a) If there exists a milestone $\mathbf{m}$ that sees $\mathbf{q}_i$ then $\mathbf{m}_i \leftarrow \mathbf{m}$
   (b) else
      i. Repeat g times:
         Pick a configuration $\mathbf{q}$ at random in the neighborhood of $\mathbf{q}_i$
         until $\mathbf{q}$ sees both $\mathbf{q}_i$ and a milestone $\mathbf{m}$.
      ii. If all g trials failed then return NO and halt, else $\mathbf{m}_i \leftarrow \mathbf{m}$.
2. If $\mathbf{m}_{init}$ and $\mathbf{m}_{goal}$ are in the same connected component of the roadmap then return YES; else return NO.

4 Analysis

In this section we give formal analyses of two variants of the roadmap planner. We try to characterize the amount of computation that the planner must do in order to give correct answers with high probability.

The intuitive reason for the experimental success of the planners introduced above is that there usually exist many collision-free paths joining two configurations. In order to bound the running times of the planners, we have to assume that $\mathcal{C}_{free}$ satisfies some geometric property capturing the above intuition. We will propose two such properties. Our thesis is that the success of any sampling strategy will stem from a similar property.

In both analyses, we consider that the key number affecting the planner's running time is the number of milestones in the constructed roadmap. Since to generate a single milestone it might be necessary to randomly pick several (possibly, many) configurations in $\mathcal{C}$, we implicitly assume that the volume of $\mathcal{C}_{free}$ relative to the volume of $\mathcal{C}$ is not too small. If this assumption is not satisfied, any variant of the roadmap algorithm presented in Section 3.2 will behave poorly. In this case, we should probably make the additional assumption that a few free configurations are given (after all, every query will give two such configurations); then, a possible sampling strategy could be to build a roadmap by generating milestones in small regions centered at the given configurations, first, and at the newly generated milestones, next.

We give below the main results of our analyses. We refer the reader to [16] (first analysis) and to [12, 13] (second analysis) for more details and proofs.

In the rest of this section we denote the volume of a subset $\mathcal{X}$ of $\mathcal{C}$ by $\mu(\mathcal{X})$. .

4.1 The Visibility Volume Assumption

For any configuration $\mathbf{q} \in \mathcal{C}_{free}$, let $\mathcal{S}(\mathbf{q})$ consist of all those configurations $\mathbf{q}' \in \mathcal{C}_{free}$ that $\mathbf{q}$ sees.

Definition 2 *Let ϵ be a positive real. A configuration $\mathbf{q} \in \mathcal{C}_{free}$ is ϵ-good if $\mu(\mathcal{S}(\mathbf{q})) \geq \epsilon\mu(\mathcal{C}_{free})$. Furthermore, $\mathcal{C}_{free}$ is ϵ-good if all the configurations it contains are ϵ-good.*

The visibility volume assumption made here is that $\mathcal{C}_{free}$ is ϵ-good, that is, each configuration in it sees a significant portion of $\mathcal{C}_{free}$. The underlying intuition is that it is then relatively easy to pick a set of milestones that, collectively, can see all of $\mathcal{C}_{free}$. However, the assumption fails to prevent $\mathcal{C}_{free}$ from containing narrow passages through which it might be difficult to connect milestones. For example, consider the case where $\mathcal{C}$ is two-dimensional and $\mathcal{C}_{free}$ consists of two disks of equal size that overlap by a very small amount. Then $\mathcal{C}_{free}$ is ϵ-good for $\epsilon \approx 0.5$. But the probability that any milestone in one disk sees a milestone in the other disk is very small. For this reason, we use a variant of the roadmap algorithm that embeds a "complex planner". We assume that this complex planner is error-free in that it discovers a path between two given configurations whenever one exists, and reports failure when there is none. But of necessity such a complete planner must be expensive to run, and so we seek to use the complex planner sparingly. This variant of the roadmap planner invokes the complex planner in its preprocessing phase only, to improve the connectivity of the roadmap. Interestingly, in a similar way, the roadmap planner presented in [14, 15] uses the potential-field planner of Section 3.1 at the end of the preprocessing stage to improve the connectivity of the roadmap.

The preprocessing algorithm is similar to the one given in Section 3.2. The main difference is that Step 4 invokes the `Permeation` algorithm rather than the `Resample` one. The constructed roadmap contains s milestones.

`Preprocessing`:

1. $i \leftarrow 0$.
2. While $i < s$ do:
   (a) Pick a configuration $\mathbf{q}$ in $\mathcal{C}$ at random.
   (b) If CLEARANCE($\mathbf{q}$) > 0 then
      i. Store $\mathbf{q}$ as a milestone of the roadmap.
      ii. $i \leftarrow i + 1$.
3. For every pair of milestones $\mathbf{m}_1$ and $\mathbf{m}_2$ do:
   (a) If $\mathbf{m}_1$ and $\mathbf{m}_2$ see each other then insert a link between them in the roadmap.
   (b) Update the connected components of the roadmap.
4. Pick one representative milestone from each component of the current roadmap. Let V be the set of these representative milestones. Invoke `Permeation`(V) to improve the connectivity of the milestones (see below).

The result of this preprocessing is a roadmap such that any two nodes are in the same component if and only if the corresponding milestones are in the same component of $\mathcal{C}_{free}$. Step 3 may fail to find all possible links between the milestones due to the incompleteness of the connector. The `Permeation` algorithm invoked in Step 4 fixes this problem by using the complex planner to discover additional connections between milestones. Note that the `Permeation` algorithm is a last resort; hopefully, much if not all of the connectivity information should have been discovered before this step. The `Permeation` algorithm is the only preprocessing step in which the complex planner is invoked.

The query processing is handled by the following algorithm, which is similar to the `Query-Processing` algorithm of Section 3.2:

`Query-Processing`($\mathbf{q}_{init}, \mathbf{q}_{goal}$):

1. For $i = \{init, goal\}$ do:
   (a) If there exists a milestone $\mathbf{m}$ that sees $\mathbf{q}_i$ then $\mathbf{m}_i \leftarrow \mathbf{m}$
   (b) else
      i. Repeat g times:
         Pick a configuration $\mathbf{q}$ at random in the neighborhood of $\mathbf{q}_i$
      until $\mathbf{q}$ sees both $\mathbf{q}_i$ and a milestone $\mathbf{m}$.
      ii. If all g trials failed then return FAILURE and halt, else $\mathbf{m}_i \leftarrow \mathbf{m}$.

2. If $\mathbf{m}_{init}$ and $\mathbf{m}_{goal}$ are in the same connected component of the roadmap then return YES; else return NO.

This algorithm returns FAILURE (instead of NO) at Step 1(b)ii. This is the only difference with the query processing algorithm of Section 3.2. Due to the use of the `Permeation` algorithm in the preprocessing phase, both the answers YES and NO are now always correct. With some probability though, the query processing algorithm may fail to give an answer. We will choose $g = \log(2/\gamma)$ where $\gamma \in (0, 1]$ is the failure probability we are willing to tolerate during a query (see Theorem 2). Note that for each i, Step 1(a) can be implemented using s invocations of the connector, one for each milestone, and that each trial of Step 1(b)i can be implemented using s invocations of the connector.

Before analyzing the query phase in detail, let us present some performance guarantee for the preprocessing phase. Call a set of milestones *adequate* if the volume of the subset of $\mathcal{C}_{free}$ not visible from any of these milestones is at most $(\epsilon/2)\mu(\mathcal{C}_{free})$. Intuitively, if we were to place a point source of light at each milestone, we would like a fraction at least $1 - \epsilon/2$ of $\mathcal{C}_{free}$ to be illuminated.

Theorem 1 *Let $\beta \in (0, 1]$ be a positive real constant. Let C be a fixed positive constant large enough that for any $x \in (0, 1]$, $(1 - x)^{(C/x \log 1/x)} \leq x\beta/4$. If $s \geq C\epsilon^{-1} \log \epsilon^{-1}$, then preprocessing generates an adequate set of milestones with probability at least $1 - \beta$.*

Note that as ϵ increases, the requirement for adequacy grows weaker and the number of milestones needed becomes smaller. Also, clearly $C = O(\log \beta^{-1})$ suffices, so that s grows no faster than the logarithm of $1/\beta$.

Theorem 1 only says that most of $\mathcal{C}_{free}$ is likely to be visible from some milestone in the roadmap; using this property alone, we can show that queries can be answered quickly. But the adequacy of the milestones is not sufficient to imply that the roadmap is a good representation of the connectivity of $\mathcal{C}_{free}$. The use of the complex planner in the `Permeation` algorithm is inevitable to ensure a good probability that the query processing outcomes YES or NO.

Assuming that the `Permeation` algorithm can connect any pair of milestones in the same component of $\mathcal{C}_{free}$ we can show the following performance guarantee for the query processing phase:

Theorem 2 *If the set of milestones chosen during preprocessing is adequate, then the probability that the query processing algorithm outputs* FAILURE *is at most γ.*

In fact, our analysis implies that the expected number of executions of Step 1(b)i in the `Query-Processing` algorithm is at most 2.

We now turn to the description of the `Permeation` algorithm and its analysis. This algorithm must determine which milestones in V are reachable from each other. If p is the size of V,

it can do so with $O(p^2)$ invocations of the complex planner by trying it on every pair of milestones in V, but we show below that far fewer invocations may suffice.

We work with the following abstract version of the permeation problem. The input is a graph N with p vertices, consisting of k disjoint cliques. The goal is to determine this clique partition of N. The graph is presented as an adjacency matrix and the cost of an algorithm is measured by the number of entries it examines in the adjacency matrix of N. This is the *edge probe model* used in the study of evasive graph properties [24]. The vertices of N correspond to the milestones in V, and an edge is present between two vertices if the corresponding milestones lie in the same component of $\mathcal{C}_{free}$. The milestones from any particular component of $\mathcal{C}_{free}$ will form a clique in N, and there are no edges between two distinct cliques. A probe into the adjacency matrix corresponds to an invocation of the complex planner.

Let $N(p,k)$ denote the non-deterministic complexity of this problem. A non-deterministic algorithm is only required to verify that some partition into k cliques is the right partition. The non-deterministic complexity of the problem is clearly a lower bound on its deterministic and even randomized complexity.

Theorem 3 *For* $1 \leq k \leq p$, $N(p,k) = \Theta(p + k^2)$.

We now characterize the worst-case deterministic complexity of this problem, denoted $T(p,k)$. Consider the following deterministic algorithm: by probing all edge slots incident on an arbitrary vertex x, determine the neighborhood of x, say $,(x)$; let $C_x = \{x\} \cup ,(x)$, and output C_x; then, recur on the vertex-induced subgraph $G[V \setminus C_x]$. The number of levels in the recursion is k, since one of the k cliques is removed from G prior to each recursive call. The number of probes made in the process of determining each such clique is at most p. The total number of probes is $O(pk)$. The following `Deterministic-Permeation` algorithm is an iterative version of the recursive algorithm (the nodes of G are named $1, 2, \ldots, p$):

`Deterministic-Permeation`(V):

1. Mark all vertices of G as being LIVE.
2. Initialize $x \leftarrow 1$.
3. While $x \leq p$ do:
   (a) $,(x) \leftarrow \emptyset$.
   (b) For $y = x + 1$ to p do:
      i. If vertex y is marked LIVE then probe the edge (x, y) in G.
      ii. If edge (x, y) is probed and found present then mark y as DEAD and add y to $,(x)$.
   (c) Output $\{x\} \cup ,(x)$ as being a clique.
   (d) Mark x as being DEAD.

(e) Set x to the smallest numbered LIVE vertex, or $p+1$ if there are no LIVE vertices left.

By the preceding discussion, we have:

Theorem 4 *The* `Deterministic-Permeation` *algorithm correctly solves the permeation problem using $O(nk)$ probes.*

The following lower bound establishes that the `Deterministic-Permeation` algorithm is optimal. The proof uses a non-trivial adversary argument [16].

Theorem 5 *For $1 \leq k \leq p$, $T(p,k) = \Omega(pk)$.*

We now give a randomized algorithm that beats the lower bound of Theorem 5 when the sizes of the k cliques differ significantly, which is often the case in practice (when $k > 1$). The `Randomized-Permeation` algorithm labels the vertices in a random order and then invokes the `Deterministic-Permeation` algorithm.

`Randomized-Permeation`(V):

1. Permute the vertices randomly. Rename the nodes by $1, \ldots, n$, in the order of the generated list.
2. Invoke the `Deterministic-Permeation` algorithm.

Let $w_1 \geq w_2 \geq \cdots \geq w_k$ be the sizes of the cliques in an instance G arranged in a non-increasing order, where $p = \sum_{i=1}^{k} w_i$. Denote by C_i the ith largest clique in G. Define a function $u()$ on an ordered k-tuple of positive integers $n_1, n_2, \ldots, n_k$ by $u(n_1, n_2, \ldots, n_k) = \sum_{i=1}^{k} i n_i$.

Theorem 6 *The* `Randomized-Permeation` *algorithm correctly determines the clique structure and incurs an expected cost that is at most*

$$2u(w_1, w_2, \ldots, w_k) - p - k.$$

Furthermore, with high probability, the cost is at most

$$O(u(w_1, w_2, \ldots, w_k) \log p).$$

Observe that the worst case is when all w_i are equal to p/k, in which case the expected cost is $O(pk)$. On the other hand when there is one giant clique and $k-1$ cliques of size $O(1)$ the expected cost is $\Theta(p + k^2)$, which is essentially the non-deterministic lower bound.

4.2 The Path Clearance Assumption

The visibility volume assumption does not prevent the existence of narrow passages in $\mathcal{C}_{free}$. To remove the need for the "complex planner" and get closer to the roadmap planner of Section 3.2, we now consider a seemingly stronger assumption: between the two configurations given by a query, there exists a collision-free path τ that achieves some clearance ξ between the robot and the obstacles. More formally, let us parametrize τ by the arc length ℓ from the initial configuration and let L designate the path's total length, i.e., $\tau : \ell \in [0, L] \rightarrow \tau(\ell) \in \mathcal{C}_{free}$. Then $\xi(\ell) = \text{CLEARANCE}(\tau(\ell))$. Let $\xi_{inf} = \inf_{\ell \in [0,L]} \xi(\ell)$.

We consider a variant of the roadmap planner in which the preprocessing algorithm consists of the first two steps of the algorithm given in Section 4.1. Unlike the planner of Section 3.2, it does not include the resampling step. The query-processing algorithm is also simpler than in Section 3.2, in that it only checks that the initial and goal configurations see milestones in the roadmap. If any one of these connections fails, the query algorithm returns NO.

Under the path clearance assumption, any NO outcome is incorrect. Let α be the probability that we are willing to tolerate for this event. The following two theorems relate the size of the roadmap to this probability, as well as to the two parameters of the assumption, that is, length L of the hypothesized path and its clearance.

Theorem 7 *Let $\alpha \in (0, 1]$ be a positive real constant. If s is chosen such that:*

$$\frac{2L}{\xi_{inf}}(1 - a\xi_{inf}^n)^s \leq \alpha, \tag{1}$$

where a is the constant $2^{-n}\mu(\mathcal{B}^1)/\mu(\mathcal{C}_{free})$ in which $\mathcal{B}^1$ denotes the unit ball in $\mathbb{R}^n$, then the planner outputs YES *with probability $1 - \alpha$.*

Note that, for any given L and ξ_{inf}, the quantity on the left side of the above inequality tends toward zero when $s \rightarrow \infty$. It is even more important to remark that it depends exponentially on s, so that the number of milestones s needed grows no faster than the logarithm of $1/\alpha$.

The following theorem is similar to the previous one, but makes use of the clearance distribution
$xi(\ell)$ rather than just its infimum:

Theorem 8 *Let $\alpha \in (0, 1]$ be a positive real constant. If s is such that:*

$$6 \int_0^L \frac{(1 - (a/2^n)\xi^n(\ell))^s}{\xi(\ell)} d\ell \leq \alpha, \tag{2}$$

where a is the same constant as in Theorem 7, then the planner outputs YES *with probability $1 - \alpha$.*

Figure 1: Illustrative example

Using the inequality

$$1 - x \leq e^{-x}, \quad \text{for } x \geq 0,$$

we get the following easier-to-use relations:

- The bound of Theorem 7 becomes:

$$\frac{2L}{\xi_{inf}} \exp(-a\xi_{inf}^n s) \leq \alpha. \tag{3}$$

- The bound of Theorem 8 becomes:

$$6 \int_0^L \frac{\exp(-a2^{-n}\xi^n(\ell)s)}{\xi(\ell)} d\ell \leq \alpha. \tag{4}$$

4.3 Example

Consider the two-dimensional problem shown in Figure 1. Below we estimate the size of the roadmap for solving this problem using the above results. The problem is parametrized by ω, as shown in the figure. We are interested in the order of magnitude of s when ω is small.

In this two-dimensional example ($n = 2$), the walls are obstacles, but have zero thickness. Hence, $\mu(\mathcal{C}_{free}) = 1$, so that $a = \pi/4$. We also have $L \asymp 1$ and $\xi_{inf} \asymp \omega$, where $x \asymp y$ means that $C^{-1}x \leq y \leq Cy$ for some constant $C > 0$.

Estimate of s using Theorem 1: The space in Figure 1 is ϵ-good for some $\epsilon = 2\omega^2$ (clearly, because of the "box" on the left).

The algorithms are those of Section 4.1. In order to bound the probability that the query processing algorithm outputs FAILURE, while a collision-free path exists (Theorem 2), the roadmap must be adequate. According to Theorem 1, this is achieved with high probability if:

$$s \asymp \frac{1}{\omega^2} \log \frac{1}{\omega}. \tag{5}$$

Estimate of s using Theorem 7: Using (3) we can derive the following order of magnitude for s that will make the planning algorithm of Section 4.2 to return YES with high probability [13]:

$$s \asymp \frac{1}{\omega^2} \log \frac{1}{\omega}. \tag{6}$$

Estimate of s using Theorem 8: The bound given by (4) leads to choosing (see [13]):

$$s \asymp \frac{1}{\omega^2} \log \log \frac{1}{\omega}. \tag{7}$$

Discussion: Note that to answer queries with high probability it is necessary and sufficient in this example to pick a bounded number of milestones in the box, which happens with probability $\asymp \omega^2$. Hence, a tight estimate of s is

$$s \asymp \frac{1}{\omega^2}.$$

The estimates (5), (6), and (7) can thus be seen as the unavoidable quantity $1/\omega^2$ times some factor. By exploiting the fact that $\xi(\ell)$ is small only briefly, we get a factor in (7) that is smaller than in (6).

5 Conclusion and Future Work

Between complete planners, which take prohibitive time to run in large dimensional configuration spaces, and adhoc planners, which are too unreliable, there exist planners that are reasonably efficient while achieving some form of completeness. In this paper we have discussed a class of such planners. These are based on a general computational scheme presented in Section 2, which consists of randomly sampling the free space and checking pairs of sample configurations for adjacency. By choosing an appropriate sampling strategy, this scheme can be turned into a concrete planner that achieves probabilistic complexity. In Section 3 we have presented two examples of probabilistically complete planners: the potential-field and the roadmap planners. In Section 4 we have analyzed in more detail the probabilistic completeness of variants of the roadmap planner, under two different assumptions: the visibility volume assumption and the path clearance assumption. In each case, we have established a relation between the probability that the planner finds a path, when one exists, and its running time (measured in that case by the number of milestones in the roadmap). Both analyses show that the number of milestones needed grows only as the logarithm of the probability of an incorrect answer that we are willing tolerate. Moreover, this number is low polynomial in the inverse of the parameter used to measure the goodness of free space (ϵ in the first analysis, ξ_{inf} in the second).

Figure 2: Robot arm example

However, how realistic and useful are our assumptions? Clearly, in most cases, there is no pratical way to verify that they are satisfied. It is also easy to create spaces that are ϵ-good for very small values of ϵ. There even exists spaces that are not ϵ-good for any positive value of ϵ. For instance, this is the case of the space comprised between two circles C_1 and C_2 tangent at some point P, with C_2 contained in C_1. On the other hand, in many applications, we are willing to discard collision-free paths if their clearance is too small, due to various uncertainties in robot control and sensing. In those cases, a planner that would return a path with high probability, whenever there exists one that has enough clearance, would be satisfactory.[7] Remark also if there exists a collision-free path with a clearance greater than some given ξ_{inf}, it must lie in a subset of $\mathcal{C}_{free}$ that is ϵ-good for an ϵ that can be derived from ξ_{inf}. Hence, the assumption of ϵ-goodness is realistic as well: for any given value of ϵ, our analysis characterizes the probability that the planner finds a path in the subset of $\mathcal{C}_{free}$ that is ϵ-good.

But we think that much more could be done. In particular, none of the two analyses we have given considers the resampling step (Step 4) in the preprocessing algorithm of Section 3.2. On the other hand, our experiments have shown that adding this step yields a much better roadmap planner. In relation to this observation, we have empirically tested the ϵ-goodness of the free space corresponding to the setting of Figure 2 (a seven-revolute-dof planar arm among several barriers forming multiple gates) for ϵ-goodness. To do so we have picked 9,000 milestones at random and we have computed how many other milestones each milestone can see. The milestones with the "most" visibility could only see about 0.06 (i.e., 6%) of the remaining milestones, suggesting that they are 0.06-good. As many as 3.3% of the milestones could see no other milestones, and fully 22% could see 0.001 (i.e., 0.1%) or less; in other words, only about 78% of the configuration space is 0.001-good or better. On the other hand, the implemented planner, which includes the resampling step, handles queries in this setting with high reliability, after having constructed a roadmap containing 5,000 to 10,000 milestones. This number is of the same order as the number suggested by Theorem 1 for $\epsilon = 0.001$ (then, $(1/\epsilon)\ln 1/\epsilon = 6,908$). Hence, the resampling step seems to lead to better coverage of the free space.

[7] Note that the roadmap planner may nevertheless return a path with a smaller clearance.

To explain the role of the resampling step we introduce a generalization of the notion of ϵ-goodness. We say that a free configuration $\mathbf{q}$ is $(\epsilon, 1)$-good if $\mu(\mathcal{S}(\mathbf{q})) \geq \epsilon\mu(\mathcal{C}_{free})$, corresponding to our original definition of ϵ-goodness for a configuration. Next, we say a free configuration $\mathbf{q}$ is (ϵ, t)-good if $\mu(\{\mathbf{q}' \in \mathcal{S}(\mathbf{q}) \mid \mathbf{q}' \text{ is } (\epsilon, t-1)\text{-good}\}) \geq \mu(\mathcal{S}(\mathbf{q}))/2$. For $t > 1$, we say that $\mathcal{C}_{free}$ is (ϵ, t)-good if $\mu(\{\mathbf{q} \in \mathcal{C}_{free} | \mathbf{q} \text{ is } (\epsilon, 1)\text{-good}\}) \geq \mu(\mathcal{C}_{free})/2$ and every free configuration is (ϵ, i)-good for $i \leq t$. If $\mathcal{C}_{free}$ is (ϵ, t)-good for a small value of t, we can give a theoretical basis for the resampling strategy. The main idea is that single links discovered by the connector in the algorithms are now simulated using t-link paths found by resampling and connecting using the connector. This leads to a generalized definition of an adequate set of milestones, and eventually to a version of Theorem 2 in which the number of invocations of the connector is larger by a factor of 2^t. See [16] for more detail.

We believe that future research should develop and analyze new sampling strategies, as well as combinations of strategies, under various assumptions. In particular, it would be of particularly interest to investigate strategies suitable for single-query planning problems (see Section 2) and to cases where the volume of the free space is very small relative to the total volume of the configuration space. We envision that randomized planners will automatically select and combine sampling strategies from a library of basic strategies to closely match the characteristics of the input problems. We believe that it should be possible to ultimately build very fast planners that can correctly handle a wide range of problems with high probability.

Acknowledgments: Jérôme Barraquand has been supported by ARPA/Army contract DAAA21-89-C0002. Lydia Kavraki has been supported by ARPA/ONR contracts N00014-92-J-1809 and N00014-94-1-0721. Jean-Claude Latombe has been supported by ARPA/Army contract DAAA21-89-C0002, and ARPA/ONR contracts N00014-92-J-1809 and N00014-94-1-0721. Tsai-Yen Li has been supported by ARPA/ONR contract N00014-92-J-1809. Rajeev Motwani is supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

References

[1] J.M. Ahuactzin Larios, *Le Fil d'Ariane: Une Méthode de Planification Générale. Application à la Planification Automatique de Trajectoires*, Thèse de l'Institut Nat. Polytechnique de Grenoble, Septembre 1994.

[2] J. Barraquand and J.C. Latombe, Robot Motion Planning: A Distributed Representation Approach, *The Int. J. of Robotics Research*, 10(6):628-649, 1991.

[3] J. Barraquand, B. Langlois, and J.C. Latombe, Numerical potential Field Techniques for Robot Path Planning, *IEEE Tr. on Syst., Man, and Cyb.*, 22(2):224-241, 1992.

[4] P. Bessierre, E. Mazer, J.M. Ahuactzin, Planning in a Continuous Space with Forbidden Regions: The Ariadne's Clew Algorithm, *Algorithmic Foundations of Robotics*, K. Goldberg et al. (eds.), AKPeters, Wellesley, MA, 39-47, 1995.

[5] J.F. Canny, *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.

[6] D. Challou and M. Gini, Parallel Formulation of Informed Randomized Search for Robot Motion Planning Problems, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 709-714, Nayoga, Japan, 1995.

[7] H. Chang, T.Y. Li, Assembly Maintainability Study with Motion Planning, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1012-1019, Nagoya, 1995.

[8] J.E. Hopcroft, J.T. Schwartz, and M. Sharir, On the Complexity of Motion Planning for Multiple Independent Objects: PSPACE-Hardness of the 'Warehouseman's Problem', *Intl. J. of Robotics Research*, 3(4):76-88, 1984.

[9] J.E. Hopcroft and G.T. Wilfong, Reducing Multiple Object Motion Planning to Graph Searching. *SIAM J. on Computing*, 15(3):768-785, 1986.

[10] T. Horsch, F. Schwarz, and H. Tolle, Motion Planning for Many Degrees of Freedom - Random Reflections at C-Space Obstacles, *Proc. of IEEE Int. Conf. on Robotics and Automation*, 3318-3323, San Diego, CA, 1994.

[11] D.A. Joseph and W.H. Plantiga, On the Complexity of Reachability and Motion Planing Questions. *Proc. of the First ACM Symp. on Computational Geometry*, 62-66, 1985.

[12] L. Kavraki, *Random Networks in Configuration Space for Fast Path Planning*, Ph.D. Thesis, Rep. STAN-CS-TR-95-1535, Comp. Sci. Dept., Stanford Univ., January 1995.

[13] L. Kavraki, M. Kolountzakis, and J.C. Latombe, Analysis of Probabilistic Roadmaps for Path Planning, manuscript submitted to *IEEE Int. Conf. on Robotics and Automation*, 1996.

[14] L. Kavraki and J.C. Latombe, Randomized Preprocessing of Configuration Space for Fast Path Planning, *Proc. IEEE Int. Conf. on Robotics and Automation*, 2138-2145, San Diego, CA, 1994.

[15] L. Kavraki and J.C. Latombe, Randomized Preprocessing of Configuration Space for Path Planning: Articulated Robots, *Proc. IROS*, 1764-1771, München, Germany, 1994.

[16] L. Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan, Randomized Query Processing in Robot Path Planning", *27th Annual ACM Symp. on Theory of Computing (STOC)*, 353-362, Las Vegas, NV, 1995.

[17] L. Kavraki, P. Švestka, J.C. Latombe, and M. Overmars, Probabilistic Roadmaps for Fast Path Planning in High Dimensional Configuration Spaces, to appear in *IEEE Tr. on Robotics and Automation*.

[18] D.E. Koditschek, Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations, *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, 1-6, 1987.

[19] Y. Koga, K. Kondo, J. Kuffner, and J.C. Latombe, Planning Motions with Intentions. *Proc. of SIGGRAPH'94*, 395-408, 1994.

[20] Y. Koga and J.C. Latombe, On Multi-Arm Manipulation Planing, *Proc. IEEE Int. Conf. on Robotics and Automation*, 945-952, San Diego, CA, 1994.

[21] F. Lamiraux and J.P. Laumond, *On the Expected Complexity of Random Path Planning*, Rep. No. 95087, LAAS/CNRS, Toulouse, March 1995.

[22] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publ., Boston, MA,1991.

[23] T.Y. Li and J.C. Latombe, On-Line Motion Planning for Two Robot Arms in a Dynamic Environment, *Proc. IEEE Int. Conf. on Robotics and Automation*, 1048-1055, Nagoya, 1995.

[24] L. Lovász and N. Young, *Lecture Notes on Evasiveness of Graph Properties*, Tech. Rep. CS-TR-317-91, Comp. Sci. Dept., Princeton Univ., 1991.

[25] M. Overmars, *A random Approach to Motion Planing*, Tech. Rep. RUU-CS-92-32, Dept. of Comp. Sci., Utrecht Univ., 1992.

[26] M. Overmars and P. Švestka, A Probabilistic Learning Approach to Motion Planning, *Algorithmic Foundations of Robotics*, K. Goldberg et al. (eds.), AKPeters, Wellesley, MA, 19-37, 1995.

[27] S. Quinlan, Efficient Distance Computation Between Non-Convex Objects. *Proc. IEEE Intl. Conf. on Robotics and Automation*, 3324-3330, San Diego, CA, 1994.

[28] J. Reif, Complexity of the Mover's Problem and Generalizations. *FOCS*, 421-4127, 1979.

[29] J.H. Reif and M. Sharir, Motion Planning in the Presence of Moving Obstacles, *proc. 25th IEEE Symp. on Foundations of Computer Science*, 144-154, 1985.

[30] J.T. Schwartz and M. Sharir, On the 'Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Advances in Applied Mathematics*, 4:298-351, 1983.

[31] P. Švestka, *A probabilistic approach to motion planning for car-like robots*, RUU-CS-93-18, Comp. Sci. Dept., Utrecht Univ., The Netherlands, April 1993.

[32] X. Zhu and K. Gupta, *On Local Minima and Random Search in Robot Motion Planning*, Technical Report, Simon Fraser Univ., BC, Canada, 1993.